

# ***PROJECT FINAL REPORT***

## ***Personal Expense Tracker Application***

### ***NALAIYA THIRAN PROJECT REPORT***

***2022***

#### ***Submitted by***

*J.Niranjana thangamanickam - 611619106061*

*R.Raveen - 611619106076*

*R.Prasanth - 611619106068*

*R.Ranjith kumar - 611619106075*

***Team ID:***

***PNT2022TMID17506***

## ***CONTEXT***

### **1. INTRODUCTION**

#### 1.1 PROJECT DESCRIPTION

#### 1.2 TECHSTACKS USED

### **2. LITERATURE SURVEY**

#### 2.1 PAPERS EXISTING

#### 2.2 REFERENCES

#### 2.3 PROBLEM STATEMENT

### **3. IDEATION AND PROPOSED SOLUTION**

#### 3.1 EMPATHY MAP

#### 3.2 IDEATION AND BRAINSTORMING

#### 3.3 PROPOSED SOLUTION

#### 3.4 PROBLEM SOLUTION FIT

### **4. REQUIREMENT ANALYSIS**

#### 4.1 FUNCTIONAL REQUIREMENTS

#### 4.2 NON-FUNCTIONAL REQUIREMENTS

### **5. PROJECT DESIGN**

#### 5.1 DATA FLOW DIAGRAMS

#### 5.2 SOLUTION AND TECHNICALARCHITECTURE

#### 5.3 USER STORIES

#### 5.4 COMPONENTS AND TECHNOLOGIES

#### 5.5 APPLICATION CHARACTERISTICS

### **6. PROJECT PLANNING AND SCHEDULING**

#### 6.1 SPRINT PLANNING AND ESTIMATION

#### 6.2 SPRINT DELIVERY SCHEDULE

## 6.3 REPORTS FROM JIRA

## 7. CODING AND SOLUTIONING

### 7.1 FEATURE

#### 1 7.2 FEATURE

#### 2 7.3 FEATURE

#### 3 7.3 DATABASE SCHEMA

## 8. TESTING

### 8.1 TEST CASES

### 8.2 USER ACCEPTANCE TESTING

## 9. RESULTS

### 9.1 PERFORMANCE METRICS

## 10.ADVANTAGES

## 11.CONCLUSION

## 12.FUTURE SCOPE

## 13.APPENDIX

## **1. INTRODUCTION**

### **a. Project Overview**

Personal expense tracker is an application or a software that takes care of all the financial activities carried out by its users. Its primary aim is to help people in managing our expenses efficiently. With the help of this application, we can get a clear idea about the inflow and outflow of money which in turn makes it convenient for us to spend the money in a wise manner. It helps you track all the transactions on a periodic basis which can be daily, weekly, monthly or even annually.

Personal expense trackers prompt users to add their expenses and update the wallet based on these expenses, which will be visible to the user. To help users get a better idea, visualizations about the expenses are displayed. Moreover, users have an option to set a limit for the amount to be used for that particular month. If the limit is exceeded, the user will be alerted through an email.

### **b. Purpose**

- Helps to keep an accurate record of your money inflow and outflow.
- Provides valuable insights to manage money in an efficient manner.
- Makes it convenient for users to devise a smart and personalized budget plan.
- Generates detailed reports to give insights about profits, loss, budgets, income, balance sheets, etc.
- Facilitates companies in achieving their business goals.

## **2. LITERATURE SURVEY**

### **a. Existing problem**

- Lack of transparency in reporting the expenses
- Delayed submission of expense reports
- Data entry errors
- Excessive number of unwanted processes in the application
- Internet issues (in case of online webapps)
- Security

### **b. References**

[https://ijariie.com/AdminUploadPdf/XTRA\\_STUDENT\\_EXPENSE\\_TRACKING\\_APPLICATION\\_ijariie16372.pdf](https://ijariie.com/AdminUploadPdf/XTRA_STUDENT_EXPENSE_TRACKING_APPLICATION_ijariie16372.pdf)

The application is used to analyze and monitor their different expenses in a particular category or by distinct kinds of spending that will help to see according to market trends. This android application maintains a record of student personal expenses such as tuition, travel, hostel fee and food, etc. The conclusion can be derived using some techniques such as clustering, classification and association.

[https://www.researchgate.net/publication/347972162\\_Expense\\_Manager\\_Application](https://www.researchgate.net/publication/347972162_Expense_Manager_Application)

A mobile application is developed for the android platform that keeps record of user personal expenses, daily transactions, money lent or borrowed. It also suggests best investment options, best ongoing offers in the market in popular categories and also provides a view of the current stock market and authenticated financial news.

<https://ijarsct.co.in/Paper391.pdf>

In this application, the user can provide his income to calculate his total expenses per day and these results will be stored for each user. The application has the provision to predict the income and expense for the manager using data mining. It can be used to reduce the manual calculations of the daily expenses and keep track of the expenditure.

[https://www.researchgate.net/publication/357644872\\_An\\_An\\_droid\\_Based\\_Mobile\\_Application\\_for\\_Tracking\\_Daily\\_Expens](https://www.researchgate.net/publication/357644872_An_An_droid_Based_Mobile_Application_for_Tracking_Daily_Expens)

es The proposed system helps in planning and monitoring one's budget to identify wasteful expenditures, adapts quickly as financial situation changes, and achieves financial goals. The database (DB) stores every active data such as users' details, expense details and reports etc. The application server manages communication with the database as inputted and retrieved by the users.

<https://research.vit.ac.in/publication/family-expense-manager-application-in-android>

It is an android application that monitors a person's costs, his/her family costs and incidental costs. This application helps to monitor every day costs, settlement points of interest, general rundown, report in detail and periodic costs subtle elements. Each of the information is put in a database and can be recovered by the client and their relatives.

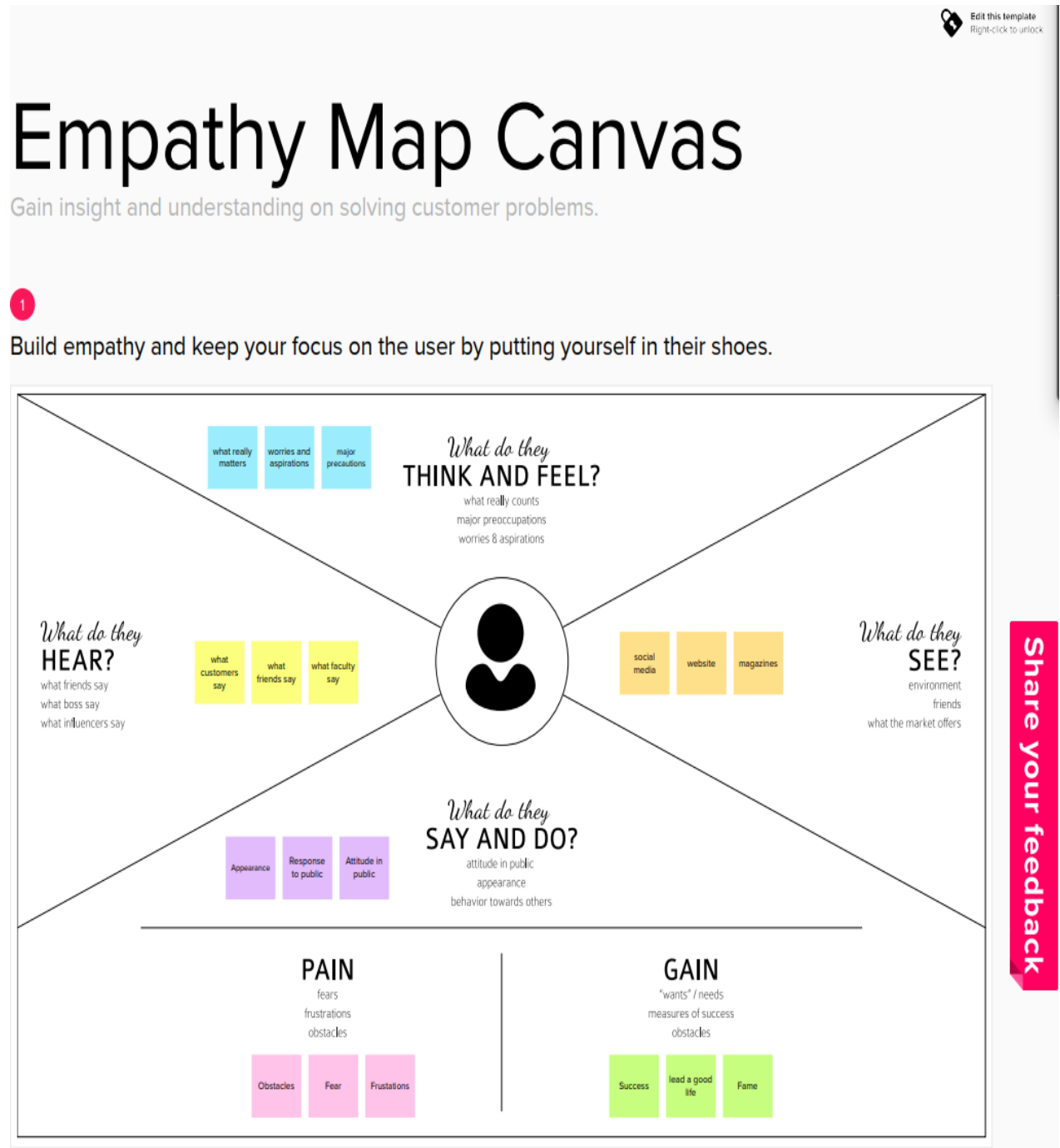
### c. Problem Statement Definition

It is tough to keep track of all the financial decisions and activities that a person makes. This problem affects all the working individuals, students and budget conscious consumers. To tackle this problem, traditional expense tracking methods were used. However, they are inconvenient, unreliable and provide limited features.

To overcome all these shortcomings, a digitized expense tracker system is being used. This system, with the help of valuable insights and visualizations, will help the users to manage all their expenses in a proper and stress-free manner

### 3. IDEATION AND PROPOSED SOLUTION

#### a. Empathy Map Canvas



## b. Ideation and Brainstorming

Template



### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare  
🕒 1 hour to collaborate  
👤 3-6 people recommended

[Share template feedback](#)

➕

**Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

➡

**Team gathering**

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

➡

**Set the goal**

Think about the problem you'll be focusing on solving in the brainstorming session.

➡

**Learn how to use the facilitation tools**

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

1

**Define your problem statement**

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

🔍

How to help the user to get tracking of monthly expenses and send alerts about spending expenses

🧠

**Key rules of brainstorming**

To run an smooth and productive session

🕒 Stay in topic.

💡 Encourage wild ideas.

⏸️ Defer judgment.

👂 Listen to others.

🗣️ Go for volume.

👁️ If possible, be visual.

2

## Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hold the pencil (click to sketch) over to start drawing!

Nirajan Thangamackam



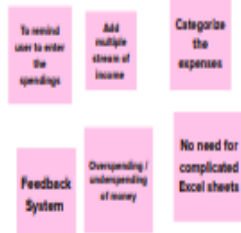
Raveen



Prasanth



Ranjith kumar



3

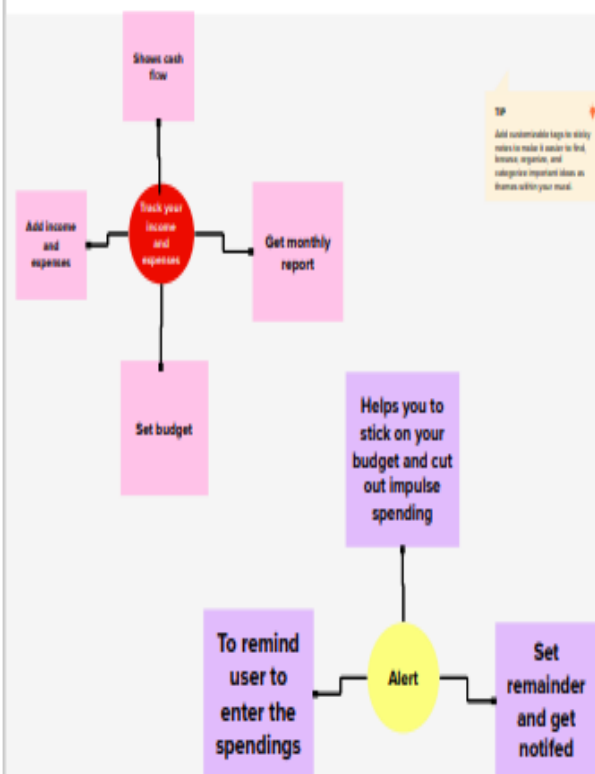
## Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, remove, organize, and categorize important ideas as themes within your mind.







#### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



#### After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

#### Quick add-ons



##### Share the mural

Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.



##### Export the mural

Export a copy of the mural as a PNG or PDF to attach to email, include in slides, or save in your drive.

#### Keep moving forward



##### Strategy blueprint

Define the components of a new idea or strategy.

[Open the template →](#)



##### Customer experience journey map

Understand customer needs, motivations, and obstacles for an experience.

[Open the template →](#)



##### Strengths, weaknesses, opportunities & threats

Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.

[Open the template →](#)



[Share template feedback](#)

### c. Proposed solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Attempting to manage the expenses of an individual in an efficient and manageable manner, as compared to the traditional way of expense tracking.
2.	Idea / Solution description	The application will be helpful for the individuals in not just managing their expenses, but also in enabling them to improve their investments.
3.	Novelty / Uniqueness	The application gives the user a chance to plan his/her monthly expenses at the start of the month. Besides this, the user gets a notification when he/she exceeds the limit that is set.
4.	Social Impact / Customer Satisfaction	With such applications, the public will start to plan their expenses better leading to their own financial stability. With more users, this application will ensure that financial state of our society improves.
5.	Business Model (Revenue Model)	Free trial for 1 month can be given to the users, so that a significant userbase is created. Following the free trial, the users can be given subscription for 3 months, 6 months or 1 year.
6.	Scalability of the Solution	Since the application takes the same set of input from all the users and does not perform many complex computations, it will be easy for us to scale the application to a larger set of users.

## d. Problem Solution Fit

### PROBLEM-SOLUTION FIT

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <ul style="list-style-type: none"><li>• Working Individuals</li><li>• Students</li><li>• Budget conscious consumers</li></ul>	<b>6. CUSTOMER CONSTRAINTS</b> <ul style="list-style-type: none"><li>• Internet Access</li><li>• Device (Smartphone) to access the application</li><li>• Data Privacy</li><li>• Cost of existing applications</li><li>• Trust</li></ul>	<b>5. AVAILABLE SOLUTIONS</b> <ul style="list-style-type: none"><li>• Expense Diary or Excel sheet</li></ul> <p>PROS : Have to make a note daily which helps to be constantly aware</p> <p>CONS : Inconvenient, takes a lot of time</p>						
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <ul style="list-style-type: none"><li>• To keep track of money lent or borrowed</li><li>• To keep track of daily transactions</li><li>• Alert when a threshold limit is reached</li></ul>	<b>9. PROBLEM ROOT CAUSE</b> <ul style="list-style-type: none"><li>• Reckless spendings</li><li>• Indecisive about the finances</li><li>• Procrastination</li><li>• Difficult to maintain a note of daily spendings (Traditional methods like diary)</li></ul>	<b>7. BEHAVIOUR</b> <ul style="list-style-type: none"><li>• Make a note of the expenses on a regular basis.</li><li>• Completely reduce spendings or spend all of the savings</li><li>• Make use of online tools to interpret monthly expense patterns</li></ul>						
Focus on JAP, lap into BE, understand HC	<b>3. TRIGGERS</b> <ul style="list-style-type: none"><li>• Excessive spending</li><li>• No money in case of emergency</li></ul>	<b>10. YOUR SOLUTION</b> <p>Creating an application to manage the expenses of an individual in an efficient and manageable manner, as compared to traditional methods</p>	<b>8. CHANNELS OF BEHAVIOUR</b> <p>ONLINE</p> <p>Maintain excel sheets and use visualizing tools</p>						
	<b>4. EMOTIONS</b> <table><tr><td>BEFORE</td><td>AFTER</td></tr><tr><td>• Anxious</td><td>• Confident</td></tr><tr><td>• Confused</td><td>• Composed</td></tr><tr><td>• Fear</td><td>• Calm</td></tr></table>		BEFORE	AFTER	• Anxious	• Confident	• Confused	• Composed	• Fear
BEFORE	AFTER								
• Anxious	• Confident								
• Confused	• Composed								
• Fear	• Calm								
Identify strong TR & EM									

## 4. Requirement Analysis

### a. Functional Requirement

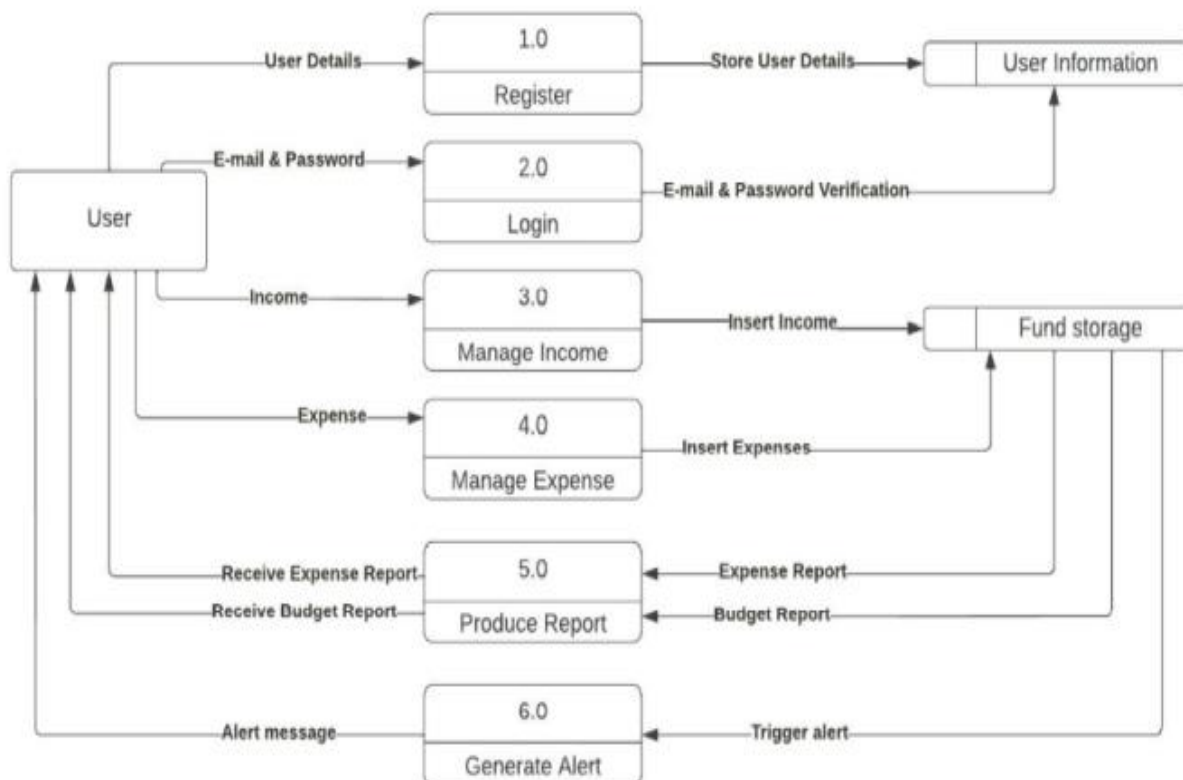
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form. Registration through Gmail.
FR-2	User Confirmation	Confirmation via OTP. Confirmation via Email.
FR-3	Add expenses	Enter day-to-day expenditure as input. Categorise the expenditure.
FR-4	Remainder mail	Sending reminder mail to the user if he/she has not filled any form of mandatory inputs.
FR-5	Creating Graphs	Graphs showing day-to-day and weekly expenses. Categorical graphs on expenditure.
FR-6	Add salary	Users must enter their salary at the start of every month, without fail.
FR-7	Export CSV	User can export the raw data of the expenditure for their own reference, in the form of a CSV file.

## b. Non-functional Requirement

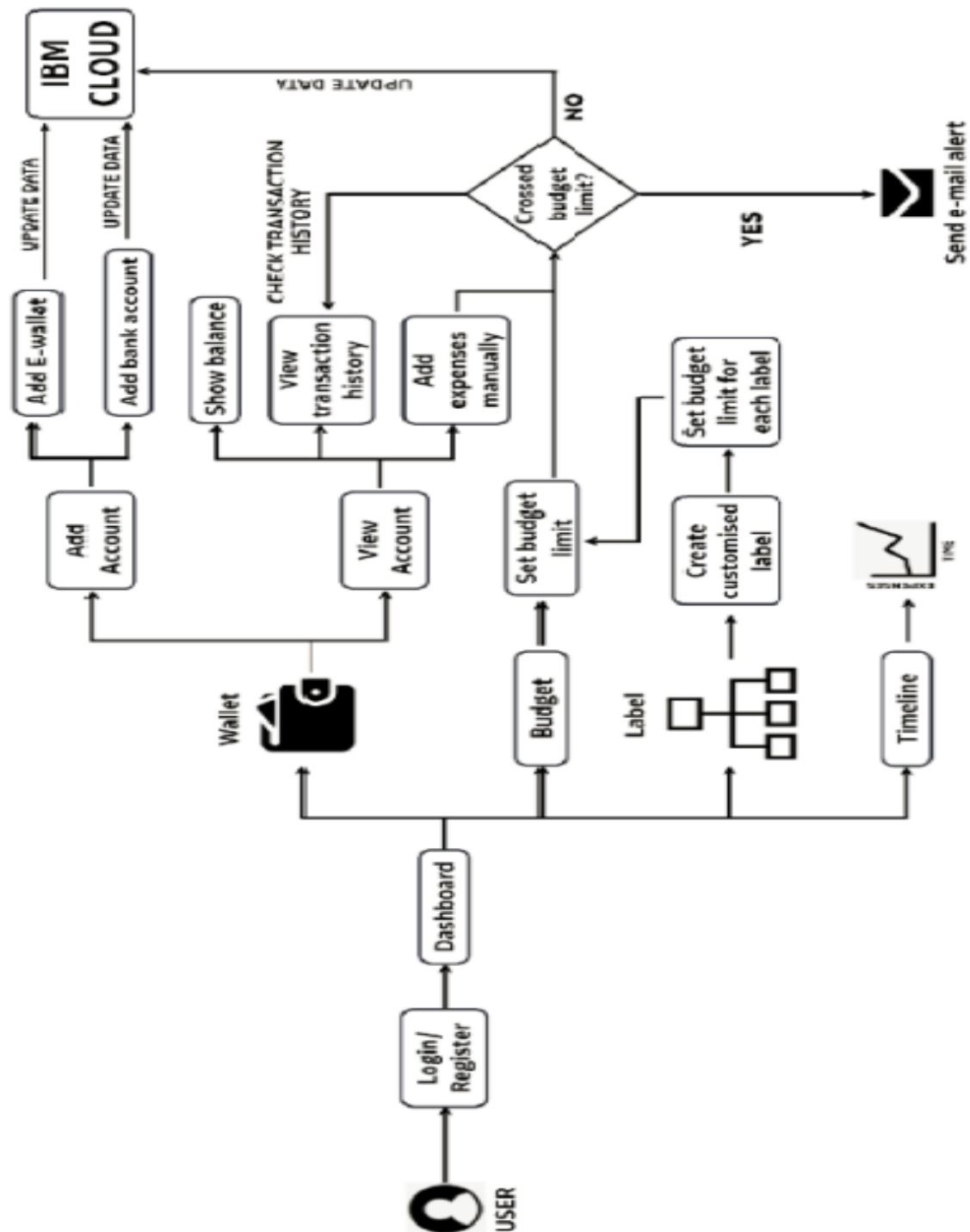
FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	A web application with an user-friendly UI and great level of accessibility.
NFR-2	<b>Security</b>	The OAuth Google sign in and email login are well secured with hashed credentials.
NFR-3	<b>Reliability</b>	Containerized service ensures that a new instance comes into picture at the time of failure.
NFR-4	<b>Performance</b>	To ensure a decent and consistent performance, the load is managed through the load balancer used with docker.
NFR-5	<b>Availability</b>	Service is always available with the presence of load balancing and multiple container instances.
NFR-6	<b>Scalability</b>	To accommodate scaling based on any requirements, Docker and Kubernetes are designed.

## 5. Project Design

### a. Data Flow Diagram



## b. Solution and Technical Architecture



## c. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
User	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can receive confirmation email & click confirm.	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application.	I can access my account / dashboard.	High	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password.	I can view my profile.	High	Sprint-1
	Dashboard	USN-4	As a user, I will be able to enter my budget, incomes and expenses.	I can change my budget plan and check income and expenses.	High	Sprint-2
		USN-5	As a user, I will be able to download monthly budget report.	I can access monthly reports.	Medium	Sprint-2
	Alert message	USN-6	As a user, I can view the alert message.	I can take actions accordingly.	Medium	Sprint-2
Administrator	Support	USN-7	As an admin, I will solve the application issues.	User can continue using the application.	Medium	Sprint-3

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	8	High	Niranjan J
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password.	8	High	Prasanth R
Sprint-2	Add Expense	USN-3	As a user, I can add the day-to-day expense to the application.	5	Medium	Raveen R
Sprint-2	Edit and Delete Expense	USN-4	As a user, I can edit and delete the previously created expense.	5	Medium	Ranjith kumar R
Sprint-3	Creating time-based filters in history	USN-5	As a user, I can see the time-based history of expenses.	8	High	Prasanth R
Sprint-3	Integrating with pie-charts for analysis	USN-6	As a user, I can view diagrammatic representation of expenses.	5	Medium	Niranjan J
Sprint-4	Enabling limit feature	USN-7	As a user, I can set monthly limit to expenses.	5	Medium	Raveen R
Sprint-4	Sending Email Alerts	USN-8	As a user, I will receive a mail if I cross a limit.	8	High	Ranjith kumar R

## b. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	08 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	15 Nov 2022



## 7. Coding and Solutioning

### 7.1 Register and Login

#### Register

```
@app.route('/register', methods = ['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        #added for bcrypt
        password = bcrypt.generate_password_hash(password)

        sql = 'SELECT * from REGISTER WHERE USERNAME = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)

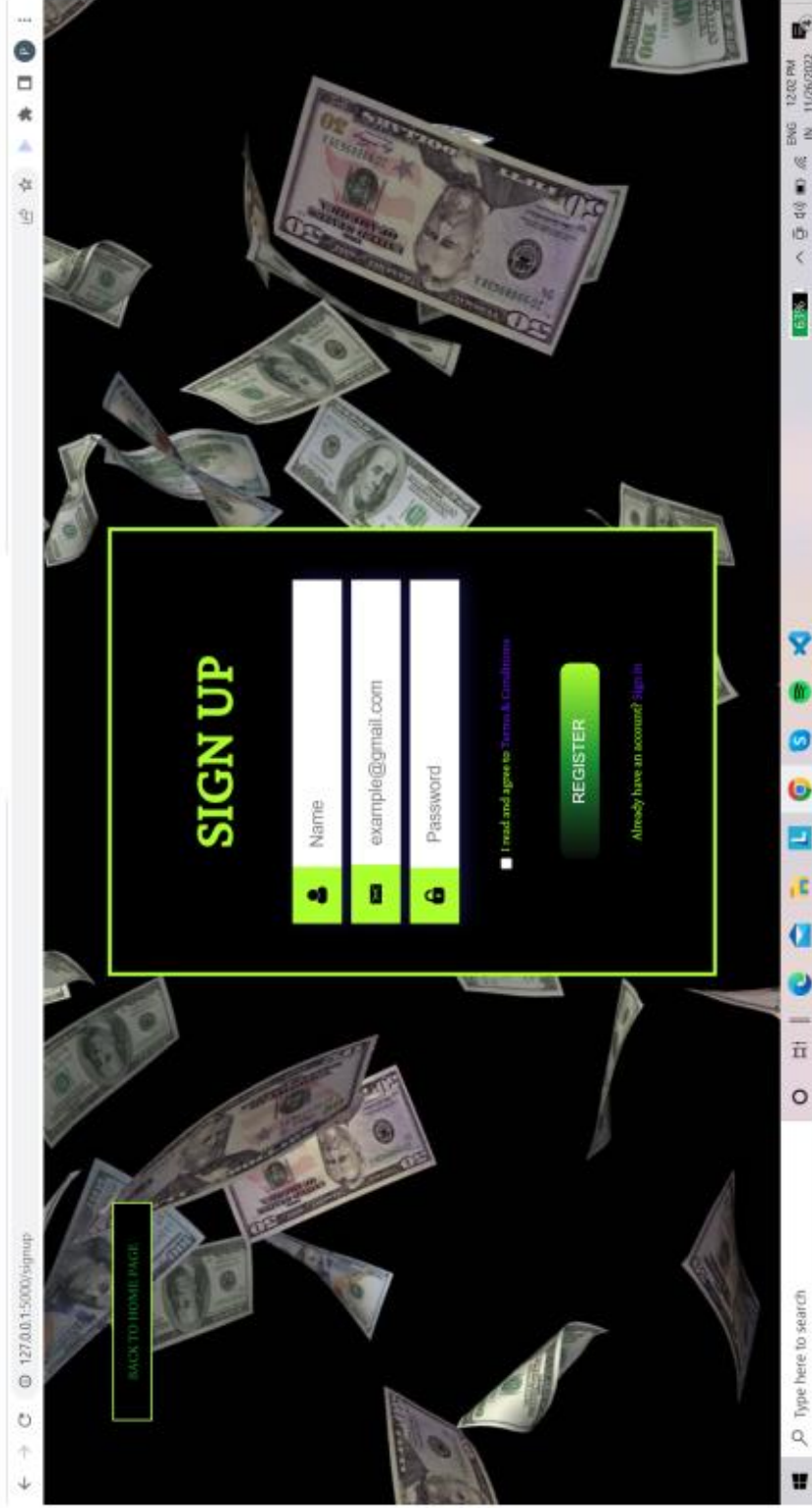
        account = ibm_db.fetch_assoc(stmt)

        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'^@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
```

```
    else:
        sql = "INSERT INTO REGISTER(USER_ID,USERNAME,EMAIL,PASSWORD) VALUES(DEFAULT,?,?,?)"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,email)
        ibm_db.bind_param(stmt,3,password)
        ibm_db.execute(stmt)

        msg = 'You have successfully registered !'
    return render_template('signup.html', msg = msg)
```

**Output:**





# Login

```
@app.route('/login', methods = ['GET', 'POST'])
def login():
    global userid
    msg = ''

    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']

        sql = 'SELECT * from REGISTER WHERE USERNAME = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)

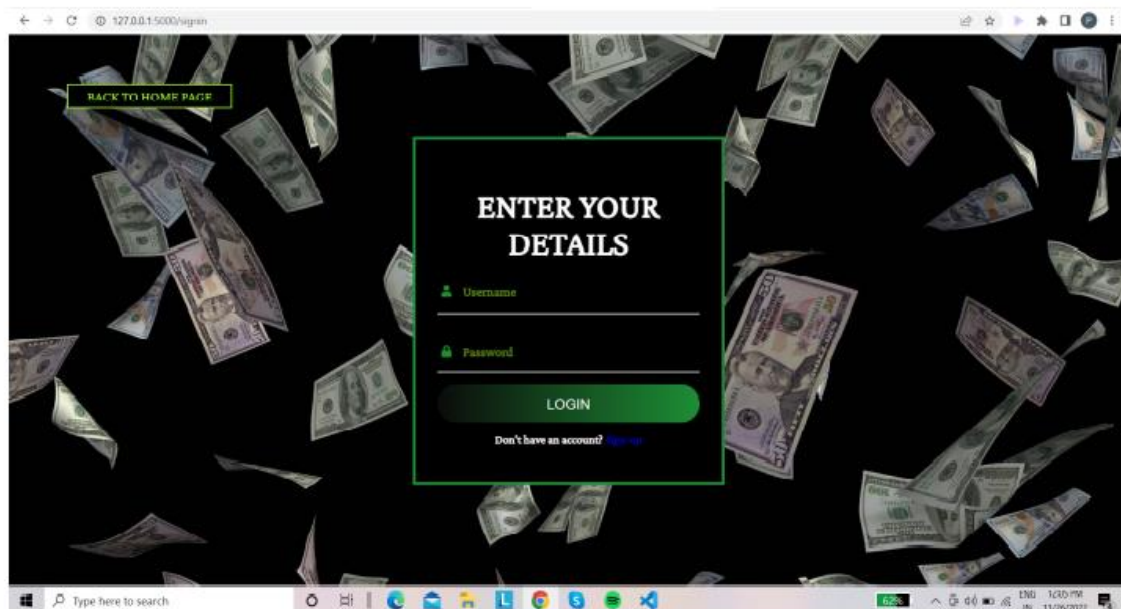
        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

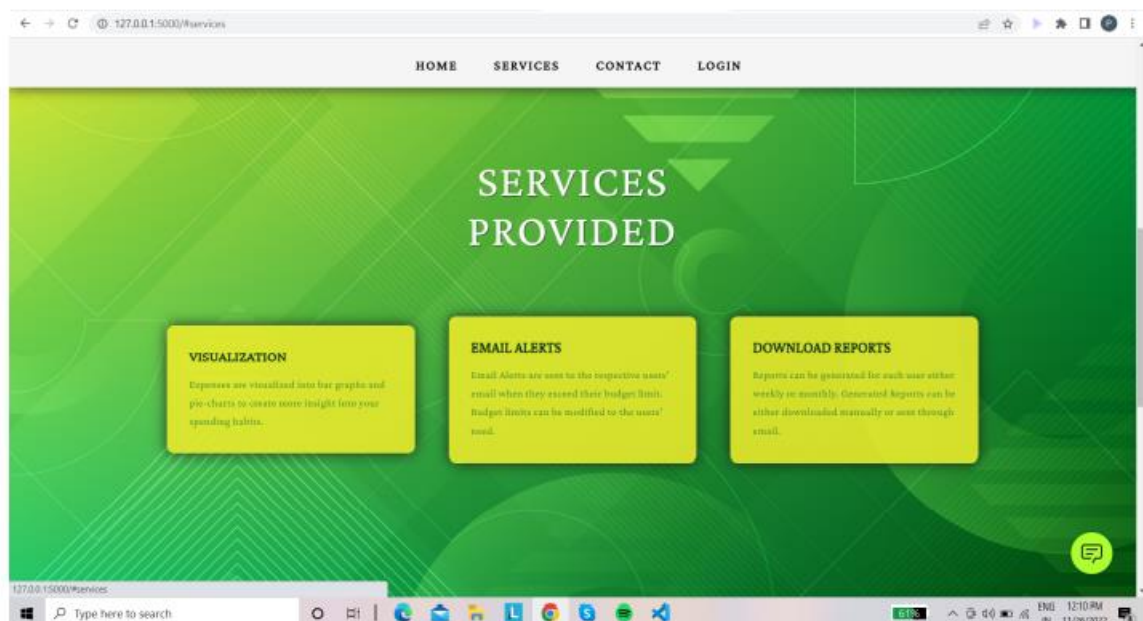
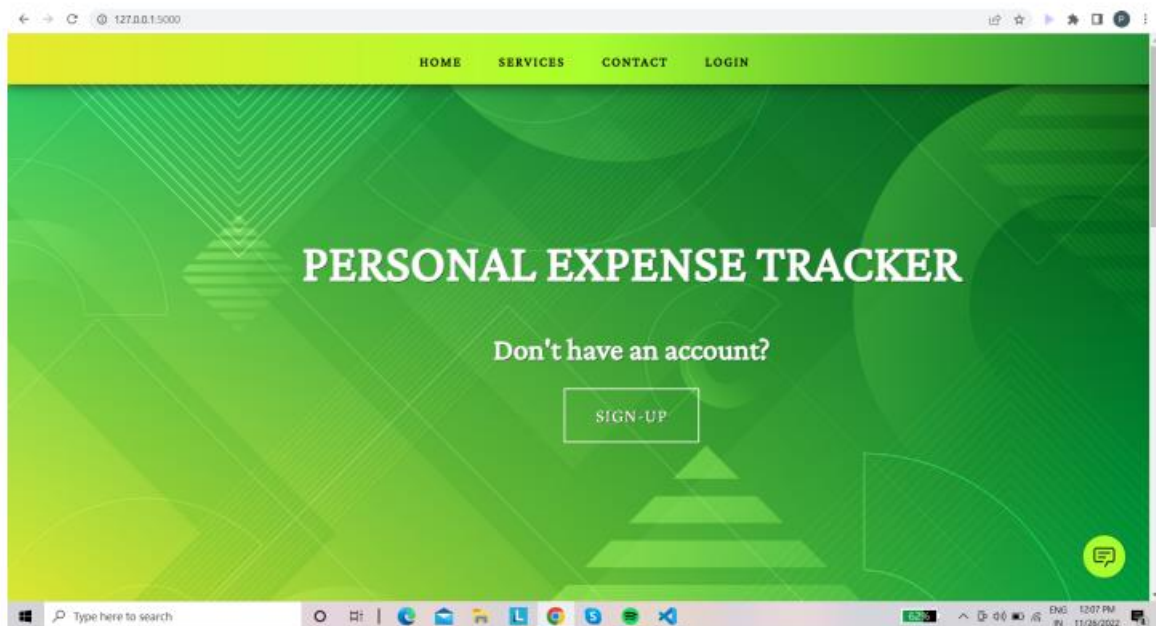
        print (account)
        #changed to add bcrypt
        if account and bcrypt.check_password_hash(account['PASSWORD'], password) :
            session['loggedin'] = True
            session['id'] = account['USER_ID']
            userid = account['USER_ID']
            session['username'] = account['USERNAME']

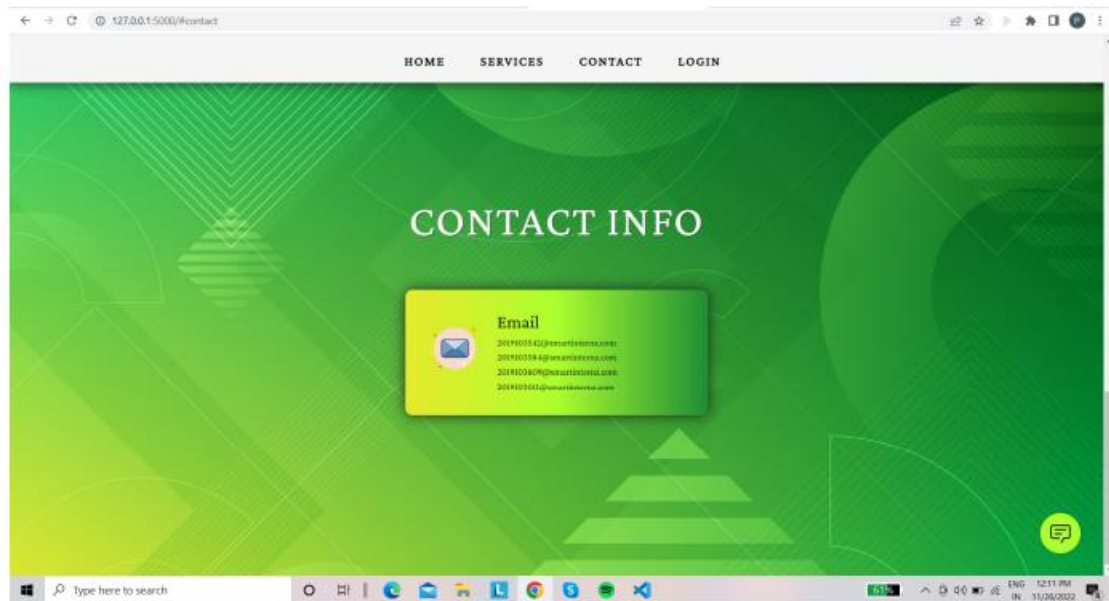
            return redirect('/add')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
```

# Output



## Home Page





## 7.2 Expenses History

```
#DISPLAY
@app.route("/display")
def display():
    if session.get("id") == None or session.get("username") == None:
        return redirect('/')

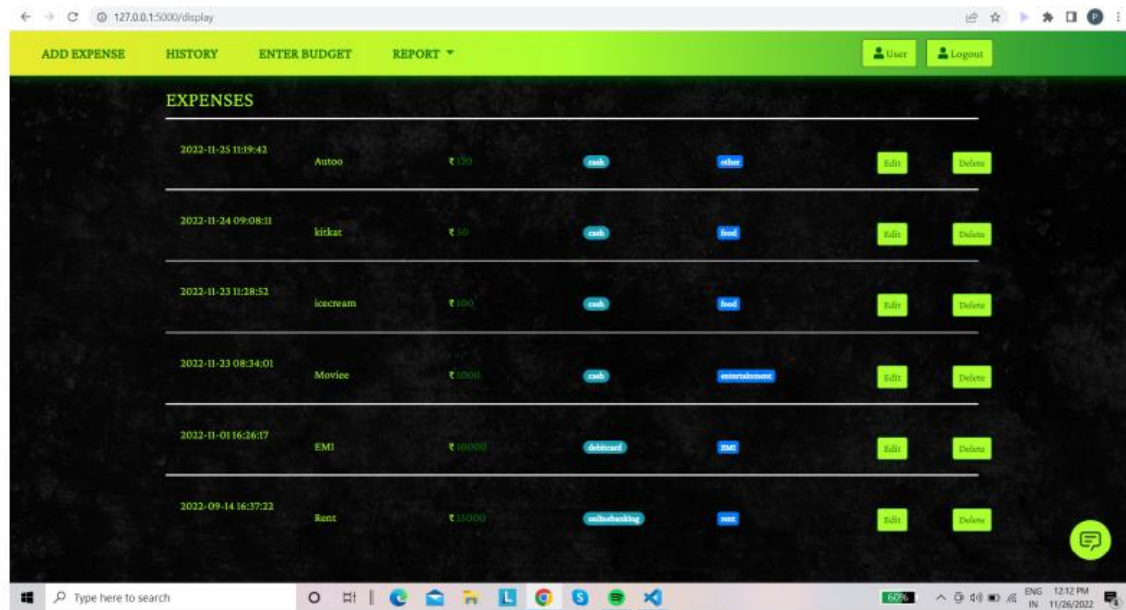
    print(session["username"], session['id'])

    id = str(session['id'])

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID = {} ORDER BY DATE DESC'.format(id)
    df = pd.read_sql(sql, pd_conn)
    expense = df.values.tolist()
    print("expense : ")
    print(expense)
    print(expense[0][3])

    return render_template('display.html', expense = expense)
```

## Output



The screenshot shows a web application interface with a green header bar containing navigation links: 'ADD EXPENSE', 'HISTORY', 'ENTER BUDGET', and 'REPORT'. On the right side of the header are 'User' and 'Logout' buttons. The main content area is titled 'EXPENSES' and displays a table of transactions. Each row includes a timestamp, a description, an amount in Indian Rupees (₹), and action buttons for 'Cancel', 'Delete', 'Edit', and 'Delete'.

Timestamp	Description	Amount (₹)	Cancel	Delete	Edit	Delete
2022-11-25 11:19:43	Autoo	₹ 130	Cancel	Delete	Edit	Delete
2022-11-24 09:08:11	kirkat	₹ 50	Cancel	Delete	Edit	Delete
2022-11-23 11:28:53	icecream	₹ 100	Cancel	Delete	Edit	Delete
2022-11-23 08:34:01	Moviee	₹ 10000	Cancel	Delete	Edit	Delete
2022-11-01 16:26:17	EMI	₹ 100000	Cancel	Delete	Edit	Delete
2022-09-14 16:37:22	Rent	₹ 10000	Cancel	Delete	Edit	Delete

## 7.3 Monthly Limit and Alert Mail

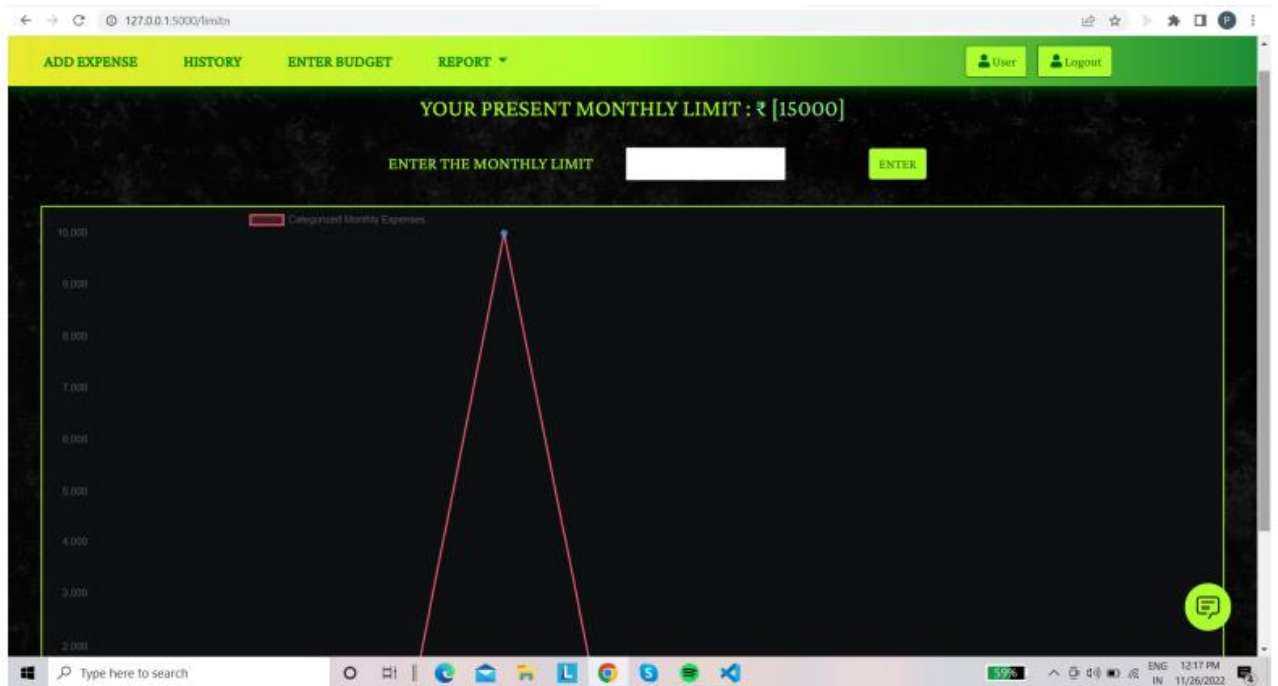
```
@app.route("/limitn")
def limitn():
    if session.get("id")== None or session.get("username") == None:
        return redirect('/')

    sql = 'SELECT BLIMIT FROM LIMITS WHERE USER_ID = {}'.format(session['id'])
    df = pd.read_sql(sql,pd_conn)
    row = df.values.tolist()

    s = 0
    if(len(row) > 0):
        s = row[0]

    rep = generateReport('Monthly')
    monthly_expense = list()
    for key,val in rep.items():
        if(key != 'expense' and key!='total'):
            print(key)
            monthly_expense.append(val[0])

    return render_template("limit.html", type="Monthly",expense_data=monthly_expense, y=s)
```



## Alert Mail - Triggered when limit is exceeded

```
def sendLimitAlert(exceded_amt):
    if(session.get('id') == None):
        return

    mailConfig()
    mail = Mail(app)

    sql = 'SELECT EMAIL FROM REGISTER WHERE USER_ID={}'.format(session['id'])
    df = pd.read_sql(sql,pd_conn)
    row = df.values.tolist()
    msg = Message('Personal Expense Tracker', recipients=[row[0][0]])
    msg.body = 'Monthly Limit Exceeded!! You have exceeded the limit by ' + str(exceded_amt)

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID={} AND MONTH(DATE(DATE)) = MONTH(CURRENT_DATE) ORDER BY DATE DESC, AMOUNT'.format(session['id'])
    df = pd.read_sql(sql,pd_conn)
    df.drop(columns=['EXPENSE_ID','USER_ID'],inplace=True)

    rep = generateReport('Monthly')
    del rep['texpense']
    df1 = pd.DataFrame(rep)
    df1.rename(columns={"t_food": "Food", "t_entertainment": "Entertainment", "t_business": "Business", "t_rent": "Rent", "t_EMI": "EMI", "t_oth": "Other"})
```

```
html = """\
<html>
<head></head>
<body>
    <h3>PERSONAL EXPENSE TRACKER</h3><br>
    <p>Monthly Limit Exceeded!!! You have exceeded the limit by Rs. {0}</p>
    <p><b>EXPENDITURE REPORT!<b><br>
        <b>Total Categorized Expenses<b><br>
        {1}
        <br><b>Monthly Expenses:<b><br>
        {2}
        <br>
        <b>Regards<b><br>
        <b>Personal Expense Tracker Team<b>
    </p>
</body>
</html>

""".format(str(exceded_amt),df1.to_html(), df.to_html())

msg.html = html
mail.send(msg)
```



# Output



spooj2002@gmail.com

to me ▾

## PERSONAL EXPENSE TRACKER

Monthly Limit Exceeded!!! You have exceeded the limit by Rs. 620

### EXPENDITURE REPORT!

Total Categorized Expenses

	Food	Entertainment	Business	Rent	EMI	Other	TOTAL
0	20	2500	0	0	1000	2100	5620

Monthly Expenses:

	DATE	EXPENSE_NAME	AMOUNT	PAYMODE	CATEGORY
0	2022-11-25 21:14:02	Movie	1000	onlinebanking	entertainment
1	2022-11-25 09:17:09	Auto	100	cash	other
2	2022-11-24 20:59:39	5-star chocolate	20	cash	food
3	2022-11-22 20:58:40	Headphones	1500	debitcard	entertainment
4	2022-11-21 21:00:34	Train ticket	1000	Pay-Mode	other
5	2022-11-10 11:11:42	American Tourister Bag	1000	cash	other
6	2022-11-01 09:06:01	Mobile Phone EMI	1000	creditcard	EMI

\*\*\*

Regards

Personal Expense Tracker Team

## 7.4 Expenses Report

```
#Generate Report Today, Monthly, Yearly
def generateReport(report_type):
    if(session.get('id') == None):
        return

    id = str(session['id'])

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    expense = []
    texpanse = []

    if(report_type == 'Today'):
        sql = 'SELECT TIME(date), AMOUNT FROM EXPENSES WHERE USER_ID = {} AND DATE(date) = DATE(NOW())'.format(id)
        df = pd.read_sql(sql,pd_conn)
        texpanse = df.values.tolist()

        sql = 'SELECT * FROM EXPENSES WHERE USER_ID={} AND DATE(DATE) = (CURRENT_DATE) ORDER BY AMOUNT DESC, DATE DESC'.format(id)
        df = pd.read_sql(sql,pd_conn)
        expense = df.values.tolist()

    elif(report_type == 'Monthly'):
        sql = 'SELECT DATE(date), SUM(AMOUNT) FROM EXPENSES WHERE USER_ID = {} AND MONTH(DATE(date)) = MONTH(CURRENT_DATE) GROUP BY DATE(date)'.format(id)
        df = pd.read_sql(sql,pd_conn)
        texpanse = df.values.tolist()

        sql = 'SELECT * FROM EXPENSES WHERE USER_ID={} AND MONTH(DATE(DATE)) = MONTH(CURRENT_DATE) ORDER BY AMOUNT DESC, DATE DESC'.format(id)
        df = pd.read_sql(sql,pd_conn)
        expense = df.values.tolist()
```

```
elif(report_type == 'Yearly'):
    sql = 'SELECT YEAR(date), SUM(AMOUNT) FROM EXPENSES WHERE USER_ID = {} AND YEAR(DATE(date)) = YEAR(CURRENT_DATE) GROUP BY YEAR(date)'.format(id)
    df = pd.read_sql(sql,pd_conn)
    texpanse = df.values.tolist()

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID={} AND YEAR(DATE(DATE)) = YEAR(CURRENT_DATE) ORDER BY AMOUNT DESC, DATE DESC'.format(id)
    df = pd.read_sql(sql,pd_conn)
    expense = df.values.tolist()

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

    elif x[6] == "entertainment":
        t_entertainment += x[4]

    elif x[6] == "business":
        t_business += x[4]
    elif x[6] == "rent":
        t_rent += x[4]

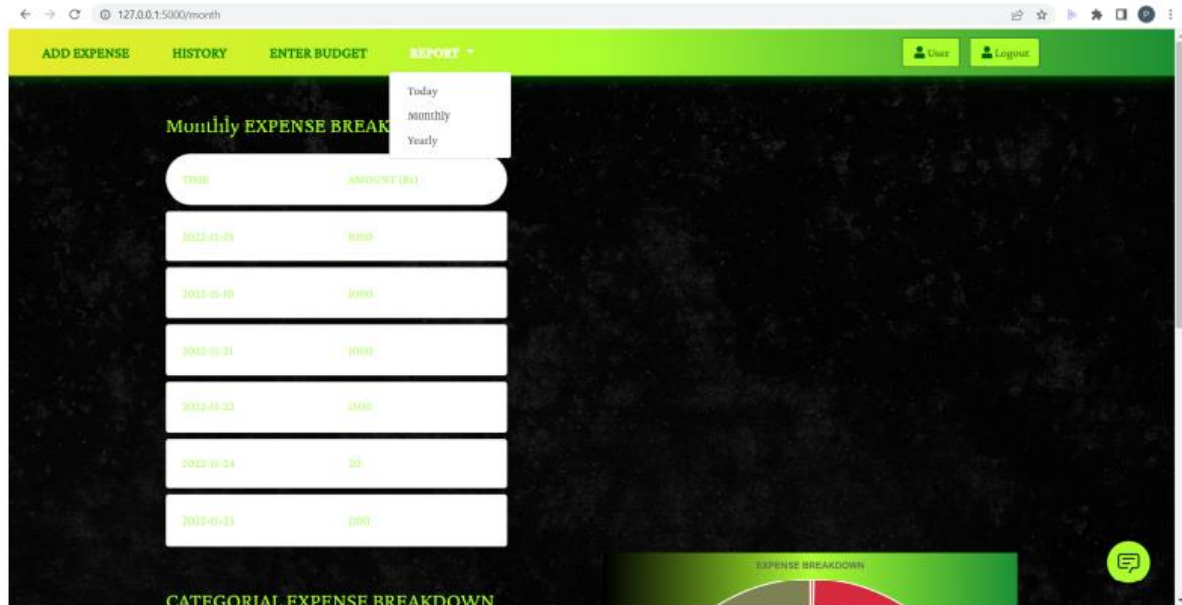
    elif x[6] == "EMI":
        t_EMI += x[4]

    elif x[6] == "other":
        t_other += x[4]

return {'texpanse':texpanse,'t_food':t_food,'t_entertainment':t_entertainment,
        't_business':t_business,'t_rent':t_rent,
        't_EMI':t_EMI,'t_other':t_other,'total':total }
```



# Output

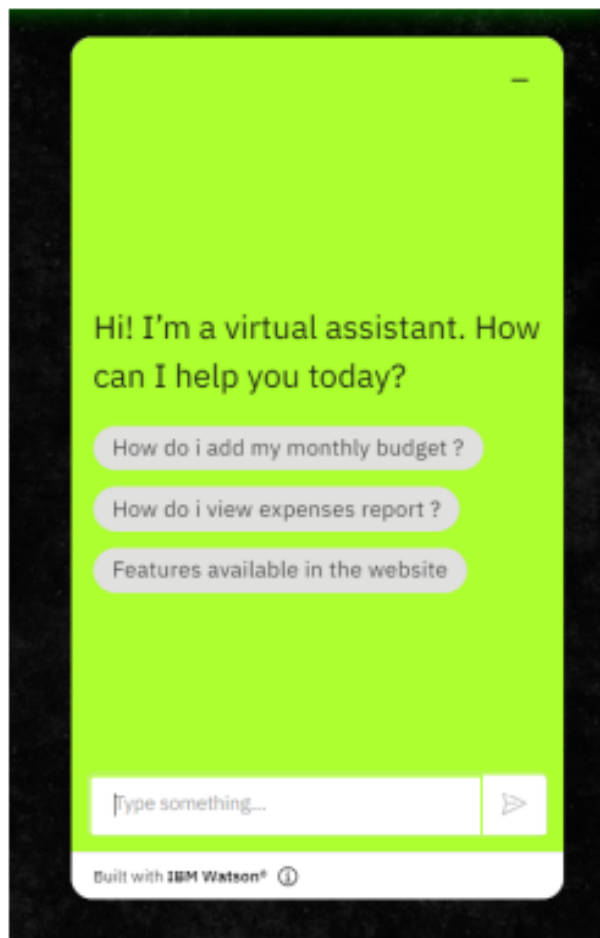
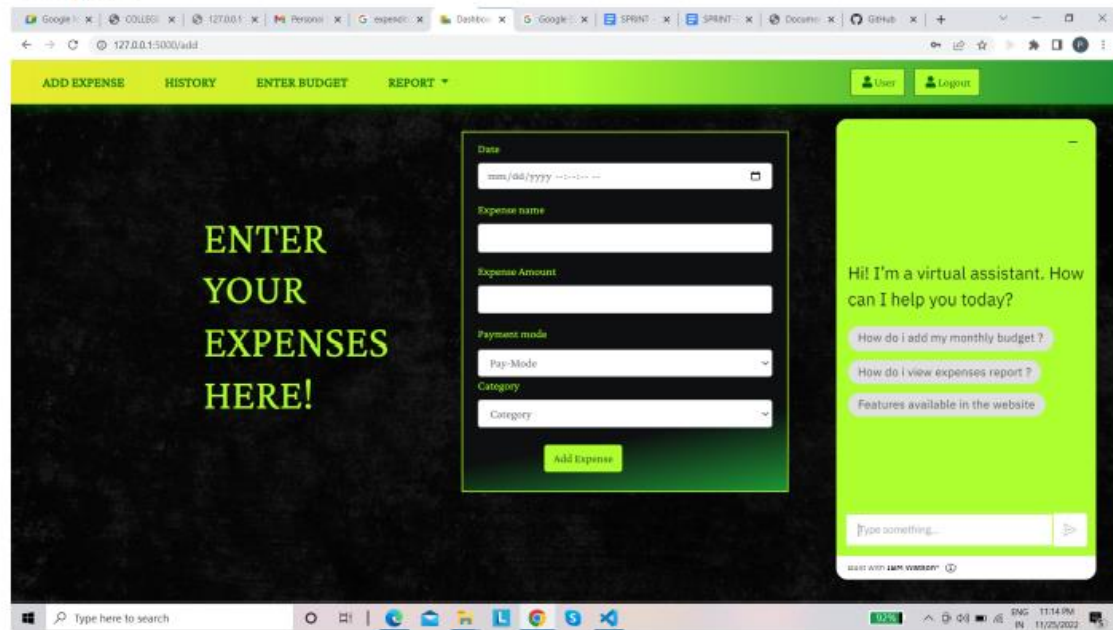


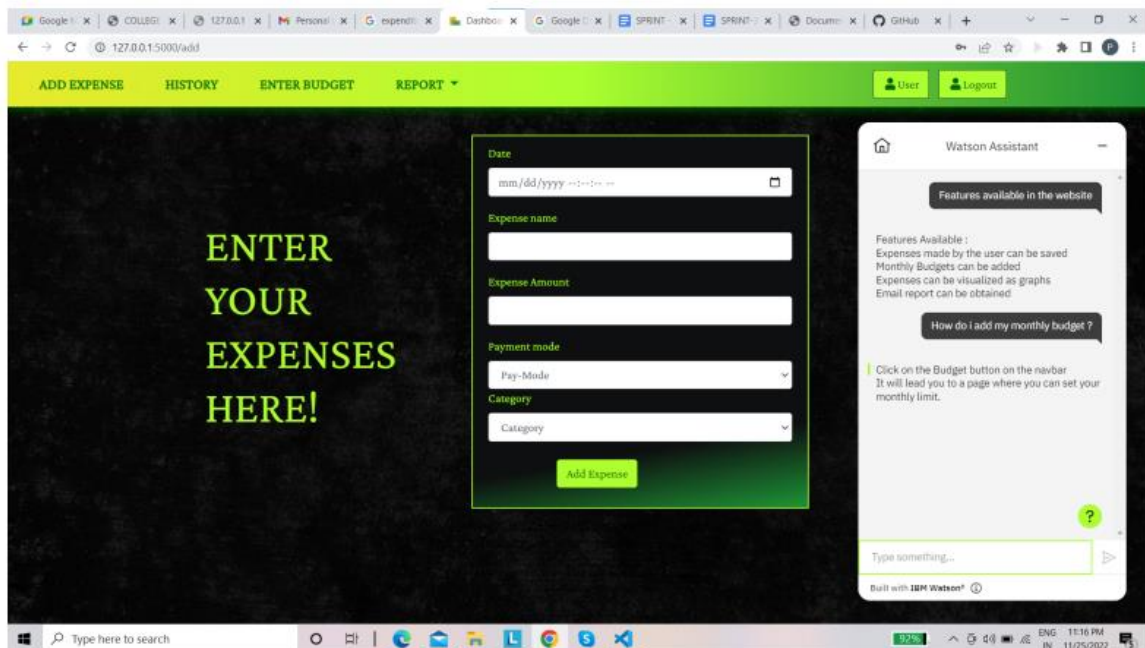


## 7.5 Watson Assistant

```
<script>
window.watsonAssistantChatOptions = {
  integrationID: "6392c642-e4e6-44a8-829a-f350772a0c1a", // The ID of this integration.
  region: "au-syd", // The region your integration is hosted in.
  serviceInstanceID: "dfebf790-1310-4e12-9378-609d293f199f", // The ID of your service instance.
  onLoad: function(instance) { instance.render(); }
};
setTimeout(function(){
  const t=document.createElement('script');
  t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
    (window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
  document.head.appendChild(t);
});
</script>
```

# Output





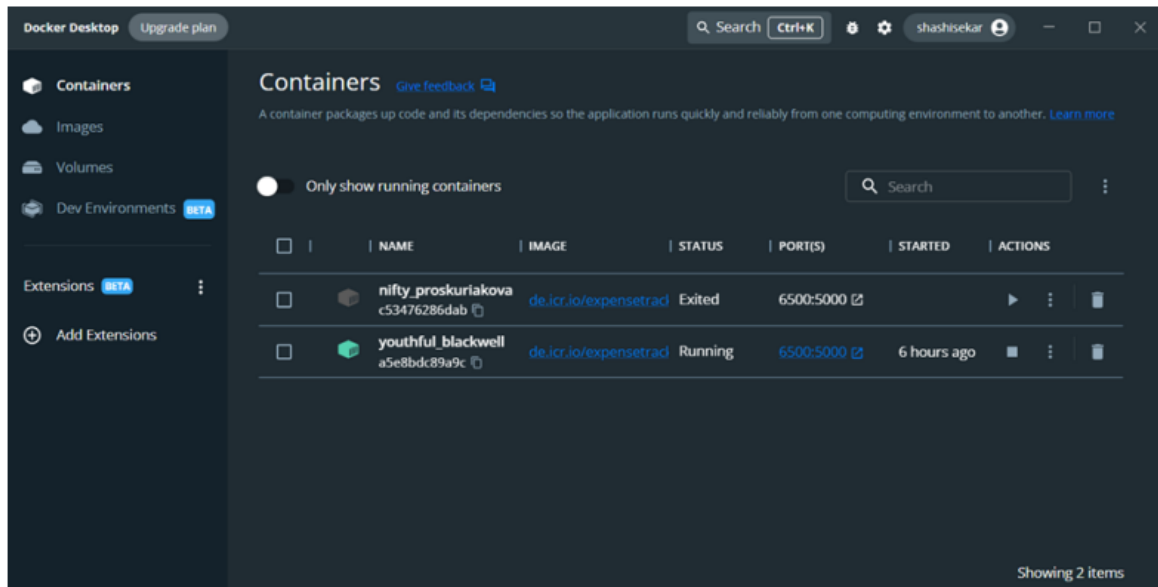
## 7.6 Docker

```

File Edit Selection View Go Run Terminal ... IBM
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

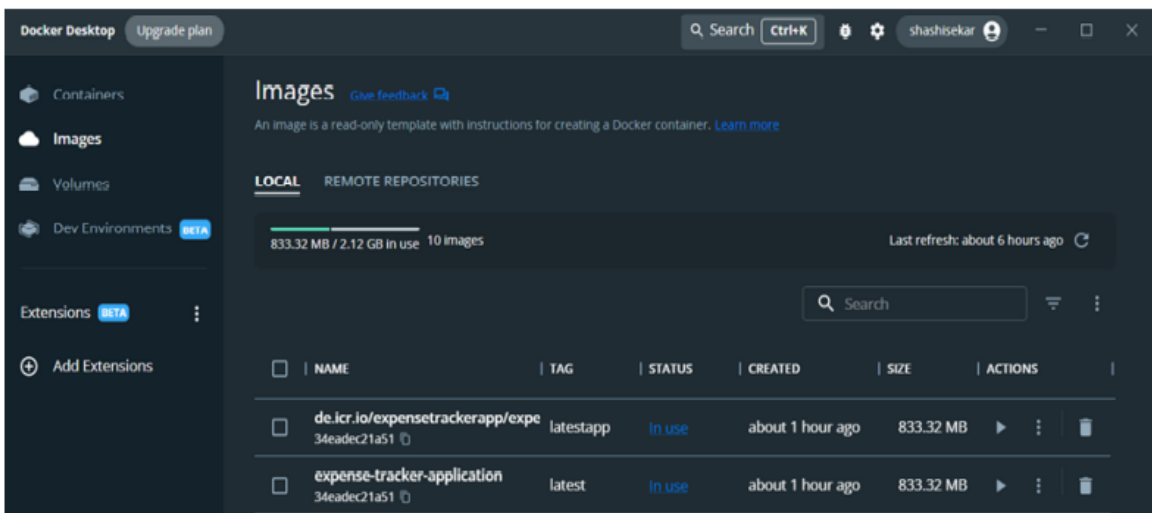
PS C:\Users\Udhana\OneDrive\Documents\Shashi - FILES\IBM\ExpenseTrackerApp> docker run -p 6500:5000 expense-tracker-application
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 117-535-145
172.17.0.1 - - [25/Nov/2022 13:49:43] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:49:43] "GET /static/css/home.css HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:49:43] "GET /static/js/home.js HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:49:43] "GET /static/images/EXPENSE1.jpg HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:49:44] "GET /static/images/expense2.jpg HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:49:44] "GET /static/images/title-icon.png HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:50:11] "GET /signin HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:50:11] "GET /static/css/login.css HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2022 13:50:11] "GET /static/js/login.js HTTP/1.1" 200 -

```



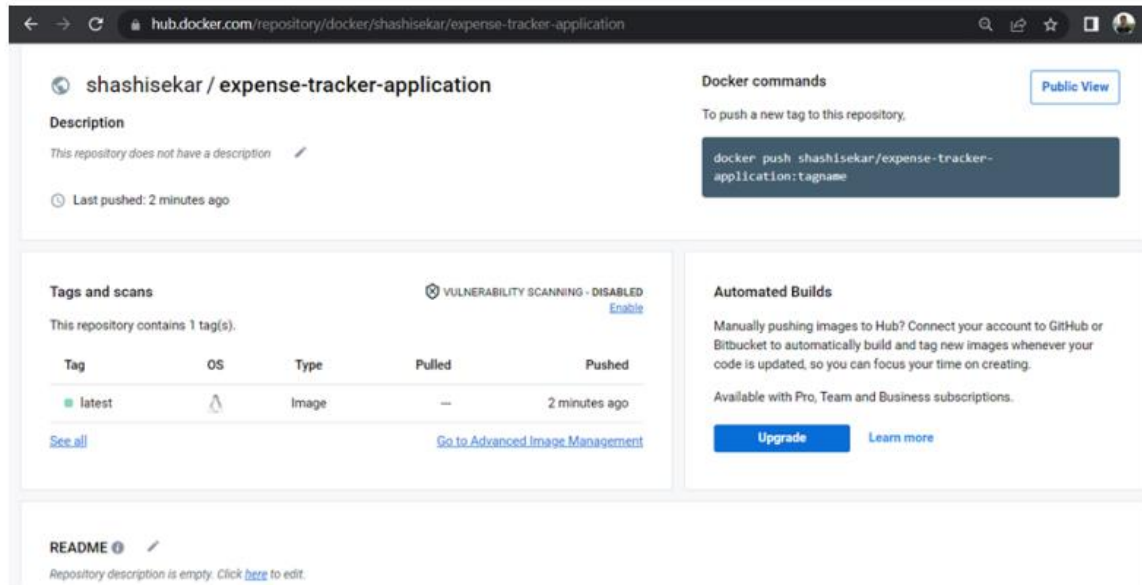
```
PS C:\Users\Dhana\OneDrive\Documents\Shashi - FILES\IBM\ExpenseTrackerApp> docker build -t expense-tracker-application .
[*] Building 724.6s (11/11) FINISHED
=> [internal] load build definition from Dockerfile                                0.5s
=> => transferring dockerfile: 32B                                              0.0s
=> [internal] load .dockerignore                                                0.6s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest                3.5s
=> [auth] library/ubuntu:pull token for registry-1.docker.io                  0.0s
=> [internal] load build context                                                0.4s
=> => transferring context: 1.67kB                                             0.0s
=> CACHED [2/5] WORKDIR /app                                                    0.0s
=> [3/5] COPY . .                                                              1.1s
=> [4/5] RUN apt-get update && apt-get install -y libc-dev python3-dev python3-pip libxml2; 619.8s
=> [5/5] RUN pip install -r requirements.txt                                   92.7s
=> exporting to image                                                            5.6s
=> => exporting layers                                                            5.4s
=> => writing image sha256:34eadec21a516135215817c8e89d75ae128143ab18a96b1c1be81f9ca35c49c5 0.0s
=> => naming to docker.io/library/expense-tracker-application                 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```





## 7.7 Docker Hub



The screenshot shows the Docker Hub interface for the repository `shashisekar / expense-tracker-application`. The page includes a description section, a Docker commands box, a tags and scans section, and an automated builds section.

**shashisekar / expense-tracker-application**

**Description**  
This repository does not have a description  
Last pushed: 2 minutes ago

**Docker commands**  
To push a new tag to this repository,  
`docker push shashisekar/expense-tracker-application:tagname`

**Tags and scans**  
This repository contains 1 tag(s).  
VULNERABILITY SCANNING - DISABLED [Enable](#)

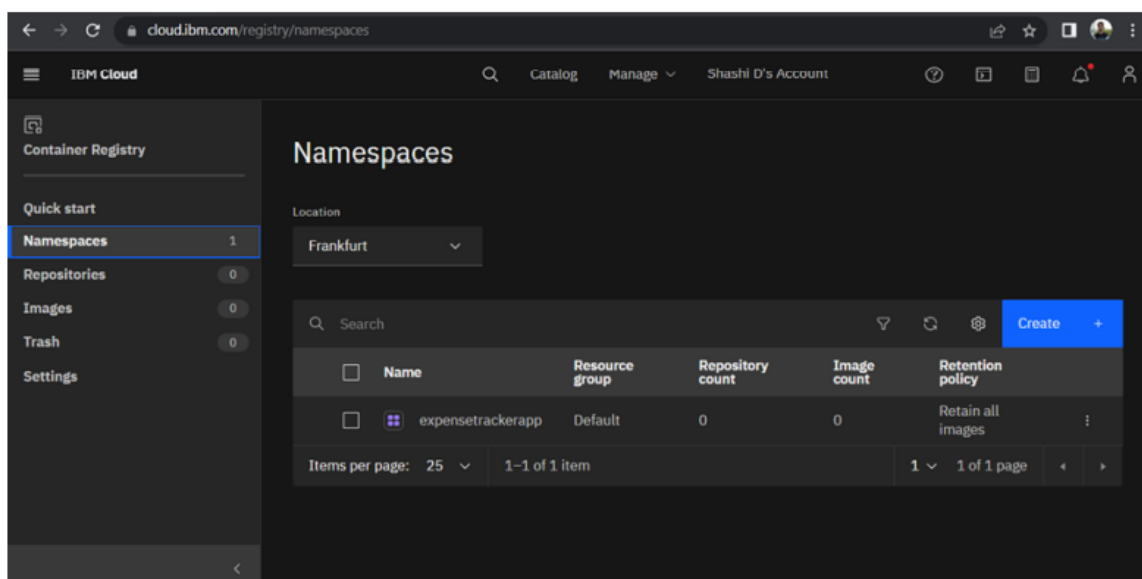
Tag	OS	Type	Pulled	Pushed
latest	linux	Image	—	2 minutes ago

[See all](#) [Go to Advanced Image Management](#)

**Automated Builds**  
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.  
Available with Pro, Team and Business subscriptions.  
[Upgrade](#) [Learn more](#)

**README**  
Repository description is empty. Click [here](#) to edit.

## 7.8 IBM Container Registry



The screenshot shows the IBM Cloud Container Registry interface for the `Namespaces` section. The page includes a sidebar with navigation options, a search bar, and a table listing namespaces.

**IBM Cloud**

**Container Registry**

**Quick start**  
**Namespaces** 1  
Repositories 0  
Images 0  
Trash 0  
Settings

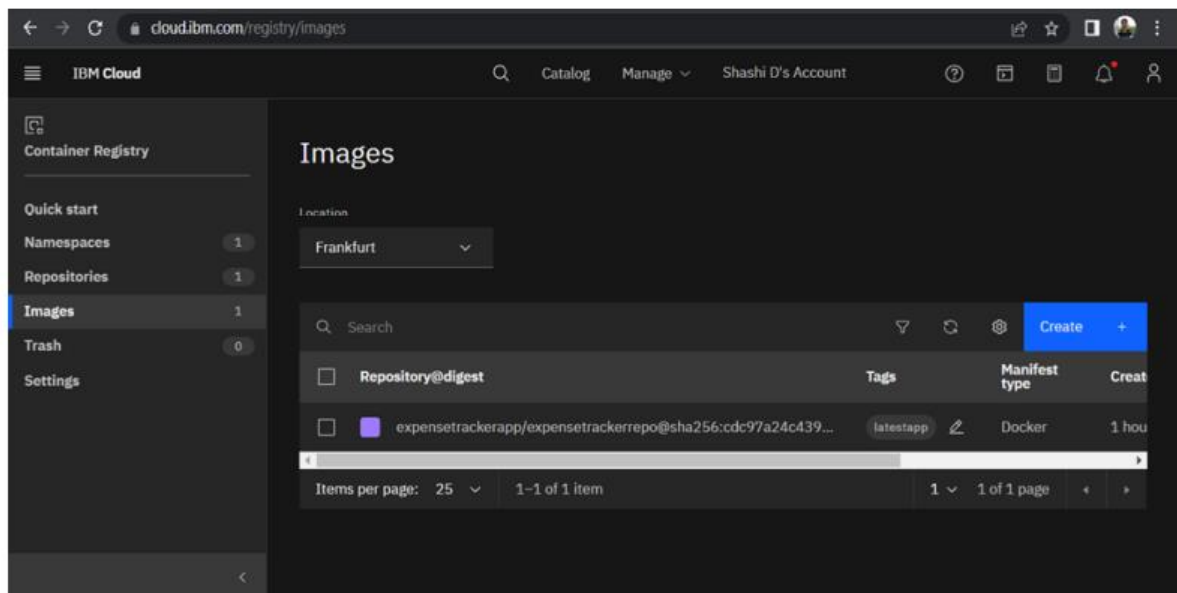
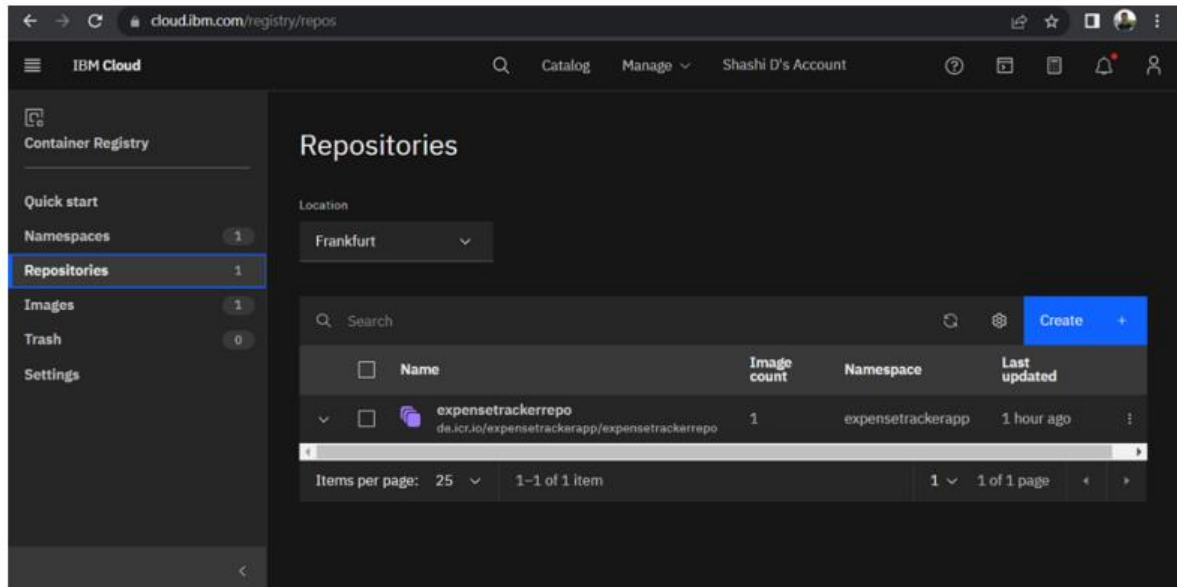
**Namespaces**

Location: Frankfurt

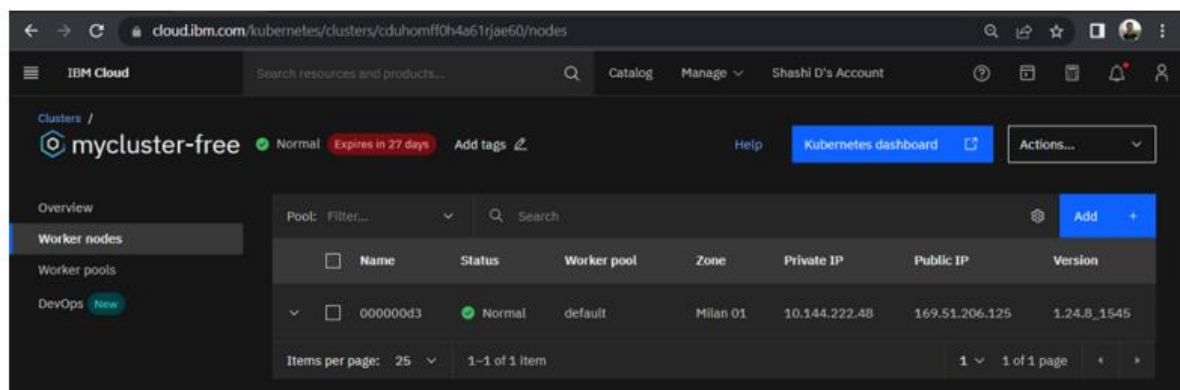
Search

Name	Resource group	Repository count	Image count	Retention policy
expensetrackerapp	Default	0	0	Retain all images

Items per page: 25 1-1 of 1 item 1 1 of 1 page



## 7.9 Kubernetes



cloud.ibm.com/kubernetes/clusters/cduhomf0h4a61rjae60/overview

IBM Cloud Search resources and products... Catalog Manage Shashi D's Account

### Node health

1 total nodes

Critical 0% Warning 0% Normal 100% Pending 0%

### Networking

Service endpoint URL

Public enabled [Copy link](#)

Private disabled [Copy link](#)

### Integrations

Logging [Connect](#) Monitoring [Connect](#) Key management service [Enable](#)

eu-de.containers.cloud.ibm.com/kubeproxy/clusters/cduhomf0h4a61rjae60/service/#/workloads?namespace=default

kubernetes default Search

## Workloads

### Workload Status

Running 1 Deployments Running 1 Pods Running 1 Replica Sets

### Deployments

Name	Images	Labels	Pods	Created
expense-tracker-app	<a href="#">Show all</a>	<a href="#">Show all</a>	1 / 1	2 hours ago

kubernetes default Search

## Workloads > Deployments > expense-tracker-app

### Metadata

Name	Namespace	Created	Age	UID
expense-tracker-app	default	Nov 25, 2022	2 hours ago	46da78f5-75a7-4050-b97e-65ff5b690920

Labels: k8s-app: expense-tracker-app

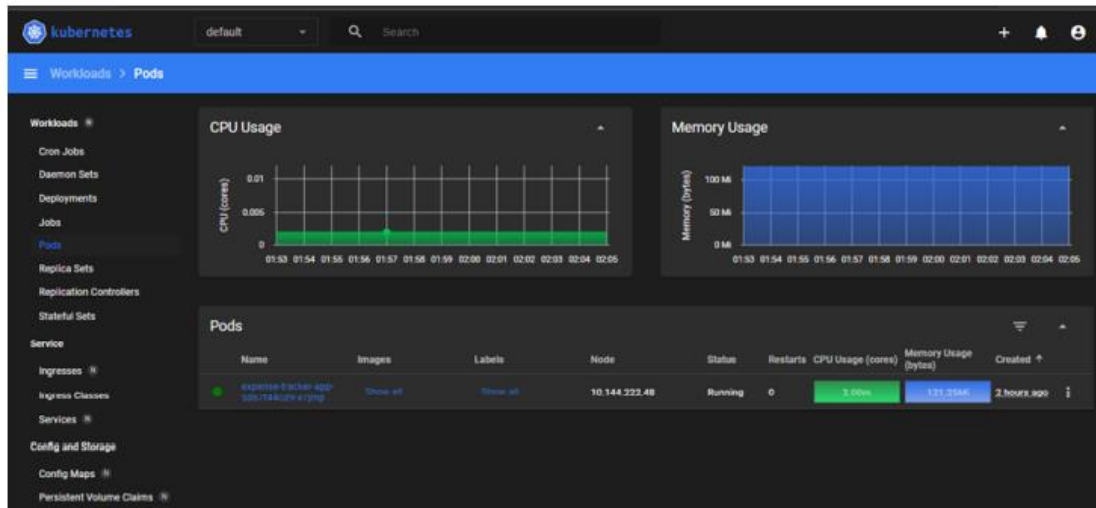
Annotations: deployment.kubernetes.io/revision: 1 [description](#)

### Resource information

Strategy	Min ready seconds	Revision history limit
RollingUpdate	0	10

Selector: k8s-app: expense-tracker-app





kubernetes default Search

Service > Services

Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Service

- Ingresses
- Ingress Classes
- Services**

Config and Storage

- Config Maps
- Persistent Volume Claims

Services

Name	Labels	Type	Cluster IP	Internal Endpoints	External Endpoints	Created
expense-tracker-app	Show all	LoadBalancer	172.21.113.244	expense-tracker-app:5000 TCP expense-tracker-app:30791 TCP	-	2 hours ago
kubernetes	Show all	ClusterIP	172.21.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	3 days ago

## 7.10 Database Schema

The screenshot shows the IBM Db2 on Cloud console interface. The top navigation bar includes tabs for Load Data, Load History, Tables, Views, Indexes, Aliases, MQTs, Sequences, and Application objects. The main content area is divided into two panels: Schemas and Tables.

**Schemas Panel:**

Name	Type	Tables
FKSB1181	User	3

Total: 1, selected: 1

**Tables Panel:**

Name	Schema	Properties
EXPENSES	FKSB1181	...
LIMITS	FKSB1181	...
REGISTER	FKSB1181	...

Total: 3, selected: 0

The screenshot shows the IBM Db2 on Cloud console interface, specifically the Table definition view for the REGISTER table. The top navigation bar is the same as the previous screenshot.

**Tables Panel:**

Name	Schema	Properties
EXPENSES	FKSB1181	...
LIMITS	FKSB1181	...
REGISTER	FKSB1181	...

Total: 3, selected: 1

**Table definition Panel:**

REGISTER

Approximate 8 rows (87.0 KB)  
Updated on 2022-11-23 20:39:42

Name	Data type	Nullable	Length	Scale
USER_ID	INTEGER	N		0
USERNAME	VARCHAR	N	255	0
EMAIL	VARCHAR	N	255	0
PASSWORD	VARCHAR	N	255	0

View data

IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

Find schemas or tables Refresh

Tables

Name	Schema	Properties
<input type="checkbox"/> EXPENSES	FKS81181	...
<input checked="" type="checkbox"/> LIMITS	FKS81181	...
<input type="checkbox"/> REGISTER	FKS81181	...

Total: 3, selected: 1

New table

Table definition

LIMITS

Approximate 3 rows (32.0 KB)  
Updated on 2022-11-23 15:19:15

Name	Data type	Nullable	Length	Scale
LIMIT_ID	INTEGER	N		0
USER_ID	INTEGER	N		0
BLIMIT	INTEGER	N		0

View data

IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

Find schemas or tables Refresh

Tables

Name	Schema	Properties
<input checked="" type="checkbox"/> EXPENSES	FKS81181	...
<input type="checkbox"/> LIMITS	FKS81181	...
<input type="checkbox"/> REGISTER	FKS81181	...

Total: 3, selected: 1

New table

Table definition

EXPENSES

Approximate 4 rows (32.0 KB)  
Updated on 2022-11-22 18:19:04

Name	Data type	Nullable	Length	Scale
EXPENSE_ID	INTEGER	N		0
USER_ID	INTEGER	N		0
DATE	TIMESTAMP	N	10	6
EXPENSE_NAME	VARCHAR	N	30	0
AMOUNT	INTEGER	N		0
PAYMODE	VARCHAR	N	30	0
CATEGORY	VARCHAR	N	30	0

View data

## 8. TESTING

### 8.1 Defect Analysis

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	1	1	2	4
Duplicate	0	0	1	0	1
External	1	0	2	0	3
Fixed	1	1	2	1	5
Not Reproduced	1	3	0	0	4
Skipped	0	2	0	0	2
Won't Fix	1	0	0	1	2
Totals	4	7	6	4	21

### 8.2 Test Case Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	2	0	0	2
Client Application	2	0	0	2
Security	3	0	0	3
Outsource Shipping	4	0	0	4
Exception Reporting	1	0	0	1
Final Report Output	2	0	1	1
Version Control	2	0	0	2

## 9. Performance Metrics

### User Acceptance Testing

NFT - Risk Assessment									
S.No	Project Name	Scope/Feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Personal Expense Tracker Web Application	New	Low	No Changes	Moderate	Yes, 2hr	>5 to 10%	GREEN	A relatively simple application meant only for expense tracking purpose.
NFT - Detailed Test Plan									
S.No	Project Overview	NFT Test approach			Risks	Approval/SignOff			
1	Login Page	1) Open the Personal Expense Tracker Application 2) Login with user Credentials			No Risks	N/A			
2	Signup Page	1) Open the Personal Expense Tracker Application 2) Enter the Details and Create a New User			No Risks	N/A			
3	Add Expense Page	1) Log in to Personal Expense Tracker Application 2) Enter all the expenses and other details.			No Risks	N/A			
4	History Page	1) Log in to Personal Expense Tracker Application 2) View the Past Expenses.			No Risks	N/A			
5	Report	1) Log in to Personal Expense Tracker Application 2) Visualization of past expenses in a day, or a month, or a year. 3) Mail or Download the respective reports			No Risks	N/A			
6	Enter Budget Page	1) Log in to Personal Expense Tracker Application 2) Budget Limits can be added.			No Risks	N/A			
7	Email alert	1) Mails are sent to the Registered user if expenses exceed the budget limit.			No Risks	N/A			
7	Watson Assistant	1) Watson assistant is enabled to handle user queries.			No Risks	N/A			
End Of Test Report									
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approval/SignOff	
	Personal Expense Tracker Web application where we can add our budget and expenses and monitor them.	1) Log in to Personal Expense Tracker web Application 2) Test for all Testcases 3) Log out.	YES	Test Passed	GO	N/A	None	N/A	

## **10. ADVANTAGES & DISADVANTAGES**

### **a. Advantages**

- (i) The application makes it easy for people to manage their expenses in an efficient way.
- (ii) The users will be notified when their expenditure exceeds the set budget limit and thus ensures to prevent overspending.
- (iii) The visualizations provided by the application helps users to get a clear idea about their expenses and thus acts as a guide for them to invest properly.
- (iv) Fraudulent activities can be identified.

### **b. Disadvantages**

- i) The process of manually entering the expenditure data can be time-consuming for the users.
- ii) Specific expenses carried out on specific days cannot be found in the application.

## **11. Conclusion**

In today's world, it is very easy for people to overspend without realizing. This can prove to be disastrous for a major section of the population who are budget-conscious. Therefore, using a personal expense tracker application can help people keep track of how much you spend every day and on what. At the end of a given period, which can be a day, month or even a year, people will have a clear picture where your money is going. This is one of the best ways to get the expenses under control and bring some semblance of order to finances of the public.

## **12. Future scope**

- Automating the process of entering the expenses can be made so as to save the time for users. Linking the users' bank accounts with the application is one way of achieving this.
- The insights provided by the web application can be made more detailed in terms of searching for expenses based on a specific date or category.

13.

### ***Project Demo Link :***

[https://drive.google.com/file/d/10qz-wBfDBjZaprJVFmR4CN1MjvRCD1Nd/view?usp=share\\_link](https://drive.google.com/file/d/10qz-wBfDBjZaprJVFmR4CN1MjvRCD1Nd/view?usp=share_link)